L16: Entry 17 of 26 File: USPT Aug 4, 1998

DOCUMENT-IDENTIFIER: US 5790664 A
TITLE: Automated system for management of licensed software

ABPL:
Methods and apparatuses are disclosed for providing a system for automatically tracking use of a software and also for determining whether the software is validly licensed and enabling or disabling the software accordingly. Exemplary systems involve attaching a licensing system module to a software application. Records of valid licenses are stored in the database maintained by the software provider. The licensing system module transparently forms a license record inquiry message. The message is transparently sent to the database over a public network, such as the Internet, to determine whether a valid license record exists in the database for the software application. The database forms and returns an appropriate response message that is interpreted by the licensing system module. The software application can then be appropriately enabled or disabled by the licensing system module. The receipt of the license record inquiry can be recorded in the database to monitor software use.

BSPR:
The present invention relates to software licensing, and in particular to a system for automated monitoring and management of licensed software.

BSPR:
It is well known that software is not purchased, but only licensed for use. Software, unlike manufactured products, can be freely copied and distributed. Hence, software providers are largely limited in their choice of control means. Unfortunately, a software license is merely a legal mechanism, and can not literally prevent illicit copying of proprietary software. A typical software license grants a permit to use the licensed software application on a particular machine and, perhaps, the generation of backup copies for personal use. A software license provides a software provider with a legal instrument against impermissible use of licensed software. However there remains no effective mechanism for preventing or monitoring illicit copying or illegal proliferation in the first place. Hence, software providers must rely on the public to not pirate software, and rely on their licensees to abstain from furnishing copies of software to friends or others. A significant amount of software piracy occurs in commercial settings. Commercial licensees are usually vigilant about license compliance. However, even the most attentive MIS manager (Management Information Systems) cannot prevent employees from copying software off of company machines for their personal use. As a result of illicit copying, software providers must adjust their prices, forcing legitimate buyers to pay higher prices to offset revenue losses.

BSPR:
There are a number of schemes designed to prevent software from being copied, or to make use of copied software unduly burdensome. These schemes, however, are largely ineffective, complex, and add to development costs. Furthermore, for every protection scheme devised by programmers, there are hackers who will diligently go about undermining them. A first line of defense is to encourage legitimate users to register their licensed software.

BSPR:
Registration of software provides a software provider with a record of a valid license. Registration typically involves filling out and mailing a registration card that is provided in an off-the-shelf software package. A user may be asked to write in the serial number of the software set, along with other pertinent

information. The defense mechanism in registration, albeit weak, is that a software provider will only render assistance and support to properly registered users. That is, a software provider will refuse to grant assistance to a user unless the user has properly registered their software.

BSPR:
In operation, a date/time checking mechanism records a date/time stamp when a software application is first brought up. Alternatively, or in addition, the date/time mechanism may start a timer when the application is brought up. The date/time stamp is compared with the system date/time information maintained by the computer to determine if the software application is to be disabled. To subvert such a system, users have been known to reset the system date and system time to prevent expiration. In response, some software providers have resorted to writing complex code schemes to disable the software in the event that the system date is tampered with. Such a security mechanism is often used to control licensed software used in a commercial setting.

BSPR:
Software sold for use in a commercial or institutional setting is frequently licensed for a predefined period of time. When such software is used on desktop computers, such computers are typically networked. The networked computers are usually connected to a file server, which file server may itself be tended by a computer management system that monitors and controls various file server groups. The file server computers act as a central location at which the desktop computers in the file server group can access files and applications. The file server also may facilitate the control of licensed software on the desktop computers. This occurs in the situation where the commercial software license is a so-called "floating license."

BSPR:
Commercial software licenses for operating a plurality of desktop computers normally are of two varieties: "fixed" or "floating." A fixed license permits a software application to run on certain designated computers (e.g., computer numbers one through five, in a ten computer file server group, are designated for the licensed software application). A floating license permits a certain number of applications to run on any number of computers at a given time. So an application operating under a floating license may be allowed to simultaneously run on no more than ten of twenty computers in a network at any given time. Licensing management software is maintained in the network file server to monitor the number of floating licenses being used.

BSPR:
Commercial software is prone to installation interruptions as it almost always requires involved enablement procedures. In accordance with regular industry practices, commercial software applications are ordinarily enabled following their installation by contacting the software provider for enablement instructions and/or enabling codes. This process is rarely instantaneous. The software provider usually confirms that the software license is proper and paid for before faxing, e-mailing, or even using regular mail, to provide a set of enabling instructions, enabling codes, or disk(s) with which to bring the application up. Consequently, the software remains disabled until additional instructions are supplied and followed, which are usually sent only after an enablement request is approved.

BSPR:
Management of floating licenses on networked computers involves two control software components: an application portion, and an authenticator portion. The application portion is nested within an application running on a desktop computer. The authentication portion is a code module contained in the file server that monitors and authorizes applications running on the desktop computers. When a user attempts to open the application software, the application portion code communicates with the authenticator code module to check to see if a floating license is available. If the maximum number of floating licenses are already being used, the software application is not allowed to open. Licensing control software also may be used to monitor defined term licenses to disable software in networked machines after license expiration.

BSPR:
If a commercial license expires, the software may be disabled, midstream,

preventing users from completing projects. Re-enablement requires contacting the software provider to purchase an additional license or extension. This may require re-execution of enablement procedures with new instructions or codes. Hence, it may take some time before the software application is up and running again, which situation can seriously inconvenience users.

BSPR:
The common shortcoming shared by all licensed software, is that it requires some form of manual intervention for registration, enablement, and/or re-enablement. Manual intervention is cumbersome and can render software useless until it is enabled or re-enabled. The paramount issue is, however, that software providers have no mechanism for monitoring and controlling the actual use, whether legitimate or illicit, of their product. Proprietary software is misappropriated on a global scale causing massive losses to software providers, which losses are inevitably passed on to legitimate licensees.

BSPR:
What is needed is a licensing system that allows software use to be monitored in an automated fashion, without user input. Moreover, a software licensing system is needed that permits a software provider to transparently control the use of licensed software.

BSPR:
The present invention addresses the foregoing problems by providing a system for automatically determining whether a software application is licensed. In accordance with the invention, a generic licensing module, or "client module," is provided that a software provider can attach to a software application. A software application having a client module attached thereto is hereinafter referred to as a "client application." In accordance with preferred embodiments of the invention, a client application loaded on a computer having access to a public network, such as the Internet, automatically reports to a computer maintained by a software provider. The client module is a program, application, or like composition of code that is preferably nested in a compiled version of a software application (i.e., to form a client application). However, the client module can, in alternative embodiments of the invention, be attached to a previously compiled software application. Whether it is referring to a program nested in, or attached to a software application, the term client module is used throughout the present disclosure.

BSPR:
A client module utilizes the public network as a means to transparently send license inquiry request messages to, and receive license inquiry response messages from, a license server maintained by a software provider. The license server has a database on which license information, or records, are stored. The license server also can record information contained in license inquiry request messages, and thereby audit use of client applications. The license record can identify a license in accordance with a hardware address, or hardware identifier of the computer, such as an IP address.

BSPR:
Alternatively, an exemplary embodiment of the invention can be used to monitor use of client applications. Operation of an exemplary system incorporating the invention for monitoring client application use involves using the client module in the client application to generate messages that are sent to the licensing server. Such messages can be sent over any public network to which a user computer, upon which the client application is loaded, is connected. For example, a message can be sent to the license server in the context of an Internet communication session. The license server tracks, or audits, the use of client applications by recording pertinent information contained in a message generated by a client module. A database can be used to store the information. A software provider or vendor can access recorded information stored in the database to generate client application use reports. Such an auditing system can be a part, or a function of, a system for enabling, validating and/or disabling licensed software (i.e., client applications). When configured as such, audit data can be derived from license inquiry request messages. Furthermore, the license record database in the license server can be used to store the collected audit data. Alternatively, a separate database can be used.

BSPR:

In a personal computer setting, an exemplary process in accordance with the invention may involve utilizing a modem, or like device, in the computer. The client module generates and sends a license validity inquiry request message to a regional or central license server maintained by the software provider. The license server contains an agent module for communicating with the client module and a database containing license records. The database in the license server is checked to see if a valid license record exists for the requesting client application and computer. If so, a message is transmitted back that allows enablement or re-enablement of the client application. The licensing server also can record information corresponding to the request in the database containing the license records, or in a different database.

BSPR:
If a license record is not found, the client application is not enabled. A menu can be presented asking whether the user would like to purchase a license, and thus enable the software. The menu may direct a user to a Web homepage where a license can be purchased, automatically open a session to such a homepage, or provide a telephone number of a sales representative or automated operator. Optionally, the user can initiate a demonstration mode of operation to evaluate the client application.

BSPR:
Operation in an exemplary process involves the client module in a desktop computer communicating upstream with an agent component in a licensing module. A client component in that licensing module communicates upstream with an agent component in a next licensing module, whose client component, in turn, communicates with a next upstream agent, and so on. This arrangement is continued upward to converge on a license server which contains an agent module. However, the license server is maintained by the software provider. Consequently, the uppermost licensing module in the institutional network communicates with the license server by initiating a connection over a public network, such as the Internet. License enablement information is supplied to the upper-most licensing module by the license server, which information is propagated back downstream via the licensing modules. The cache components in the licensing modules can be used to store license records so that license inquiries can be addressed without having to forward the validation inquiry request messages to the license server.

BSPR:
In accordance with exemplary embodiments of the invention, if a client application does not receive enablement information, the client application is not enabled, or is disabled. Hence, any software application that contains a client module accordance with the invention, can be automatically enabled, or disabled. Furthermore, use of client module equipped applications can be tracked. Such a system allows software to be freely distributed while ensuring that a license is taken for its use, or at the very least, ensuring that the use of the software can be tracked.

DRPR:
FIG. 6 depicts an additional exemplary embodiment of the invention wherein multiple software application licenses are managed; and

DEPR:
FIG. 1 depicts a personal computer system in accordance with an exemplary embodiment of the invention. The system includes a personal computer 100 that has a client application 103 residing on a hard drive 104. The client application 103 is comprised of a software application 102 and a client module 108. The computer 100 includes a modem 106. The client module 108 operates to enable or disable the software application 102 pursuant to a response from a license server 110 in the context of license validity inquiries. The license server 110 contains a database 112 having license records recorded thereon, and an agent module 114 that communicates with the client module 108. The licensing server 110 is typically maintained by the software provider who developed the software application 102. Alternatively, the license server 110 can be maintained by a contracted service provider. In a preferred embodiment, the client module 108 and the agent module 114 communicate over the Internet 116. However, the client and agent can communicate over any public network. As used herein, the term public network encompasses not only networks that are freely available to the public, generally, but also any private network which can be subscribed to. The depiction of the client module 108 is merely for descriptive and illustrative purposes. The client

module 108 can be code nested within the software application 102.

DEPR:
In accordance with an exemplary embodiment of the invention, the client module 108 automatically initiates a process to determine whether the software application 102 is validly licensed. This can happen each time the client application 103 is brought up. The licensing module 108 operates transparently and utilizes the modem 106 to form a connection with the licensing server 110. Once the connection is made, the client module 108 sends a license validity inquiry request message to the license server 110. The agent module 114 receives the request and queries the database 112 to determine whether a license record exists that corresponds to the client application 103 and computer 100. The license server 110 also can record relevant information contained in the license validity inquiry request message to audit the use of client application.

DEPR:
Once the connection is confirmed (step 206), the client module 103 forms a license validity inquiry request message (step 208). The request message may contain information such as the application name, the application version number, a date/time stamp, the name of a license server 110 (if several license servers are maintained by the software provider), and a hardware identifier, such as the IP address of the computer 100. After formation, the request message is sent to the license server 110 (step 210) over a public network*. The agent module 114 in the license server 110 forms a query (step 212) to determine whether a corresponding license record is stored in the database 112 (step 214). The agent module 114 also can record audit information from the request message (step 213). If the query locates a record of a license for the request, a response message is returned having a license ID field comprising a pointer to the location of the license record in the database 112 (step 218). If the query does not locate a record of a license for the request, a response message is returned having a null indication in the license ID field (step 216). The response message is returned to the client module 108 (step 220) after which the Internet connection is closed (step 222).

DEPR:
The client module 108 investigates the response message to determine whether the license ID field contains a license ID (step 224). If the license ID field is null, the client module 108 fails to enable the software application, or disables it (step 226). The client module 108 may then prompt the user with any variety of messages (step 227). For example, the user may be prompted to assess whether a demonstration period of operation would be acceptable. If so, this information can be recorded in the client module 108 and be passed upstream in the context of a next validity inquiry request message. The server 110 will record this information in the database 112. Alternatively, the user can be prompted to contact a sales representative or automated operator to purchase a license, or directed to a Web homepage where a license for the software application can be purchased. In the event of a license purchase, the database 112 can be automatically updated to record the license. Thereafter, a validity check will find a license record and allow the client application 103 to be enabled.

DEPR:
In the event that no license is found, several response options are available which vary according to the requirements of, and discretion of a designer of the software application 102. As previously mentioned, a response can be to provide the user with a phone number through which a software license can be purchased, or to direct the computer user to a Web homepage maintained by the software provider. Alternatively, the client module 108 can directly initiate a session with the Web server 118 that supports a homepage through which the user can purchase a license. A first screen on such a homepage can prompt the user to indicate whether the purchase of a full license would be desirable, or whether a demonstration period is preferable to evaluate the application. If neither of these options is selected the session is terminated. If the user opts to take a license, the user can be prompted with questions asking which features in the software application are to be enabled (the price of the license can be adjusted accordingly). The session can conclude with the presentation of a payment screen inviting the user to enter credit card information, or to call a sales representative in order to supply payment information.

DEPR:

If credit card information is supplied in the homepage session, it can be gathered using the system disclosed in the U.S. Pat. application Ser. No. (BDSM Attorney Docket No. 025553-014) entitled: "System for Securely Storing Information Received Over a Public Network," by Coley and Wesinger, filed on Feb. 6, 1996, and incorporated herein by reference in its entirety. Once the credit card information is entered, a response message can be sent to the client module 108 temporarily enabling the software application 102. The database 112 can then automatically updated with a license record. If a credit card turns out to be invalid, the license server database 112 can be updated accordingly by removing the license record and thereby disabling the software pursuant to a next inquiry.

DEPR:
The exemplary inventive system described above allows client applications (i.e., software application having client modules) to be freely distributed while reasonably ensuring that they are, or will be, licensed if used. Any software application having a licensing system client module attached will not operate unless and until the license system client module receives authority to enable the software application. Such a system allows global proliferation of the software, even in the form of a copy. However, such widespread use of client applications may result in the license server 110 being inundated with validity request message traffic. A dedicated license server can be set up to handle all of the license inquiry traffic for a particular software application. Alternatively, some form of traffic management can be invoked.

DEPR:
Traffic management can take many forms. It can involve establishing regional license servers according to a geographic arrangement that permits efficient response to any licensing inquiry request messages. A client application initialization process can be used wherein a user enters the location (e.g., zip code, city, and country). This information can be used by the client module to select an appropriate autodial telephone number whereby a nearest software provider license server can be accessed.

DEPR:
FIG. 4 depicts a commercial network system in accordance with an exemplary embodiment of the invention. Desktop machines 400 are organized in file server groups. The file server groups are administered by file server computers 402 through networks 404. The file server groups can, for example, serve various design teams in a research and development facility of a corporation. The file servers 402 in the R&D facility are, in turn, tended by a minicomputer 406. The minicomputer 406, and minicomputers 408 and 410 at other facilities (e.g., manufacturing and sales) are networked under a main computer 412 located, e.g., at the headquarters of the corporation. In accordance with an embodiment of the invention, each desktop computer 400 contains a client module for monitoring one or more client applications. The client modules in the desktop computers 400 communicate upstream with licensing modules contained respective file server computers 402. The licensing modules in the file server computers 402 communicate with a licensing module in the minicomputer 406, which licensing module, in turn, communicates with a licensing module in the main computer 412 at the corporation headquarters. The licensing module in the main computer 412 uses a public network, such as the Internet 414, to communicate with a license server 416 maintained by a software provider who developed the software application(s) on the desktop computers 400. Main computers 413 and 415 at other corporations or institutions also can communicate with the license server 416 to communicate license inquiry and response messages.

DEPR:
A representation of the network scenario depicted in FIG. 4, illustrating licensing system components in accordance with an exemplary embodiment of the invention, is shown in FIG. 5. Various network computers are depicted in symbolic form to assist in illustrating the components involved in the exemplary embodiment of the invention. Desktop computers 500 contain software applications 514 having licensing system client modules 516 attached thereto. The desktop computers 500 are tended by group file servers 502 on networks 504. Each of the group file server computers 502, minicomputers 506, 508 and 510, and a main computer 512 contain a licensing module. A licensing module comprises an agent component 518, a cache memory component 520, and a client component 522. The license server 526 maintained by the software provider contains an agent module

524. For any of the licensing modules in the intermediate computers between the desktop computer 500 and the license server 526, the licensing module's client component 522 communicates with the agent component 518 of an upstream licensing module, or with the agent module 524 of the license server. The licensing module's agent component 518 communicates with a downstream licensing module's client component 522, or a client module 516 in a desktop computer 500. Communication between the upper-most licensing module in the internal network (i.e., licensing module 512) and the agent module 524 in the license server 526, is conducted over a public network, such as the Internet 528.

DEPR:
An audit function can be implemented in a networked embodiment of the present invention in a number of ways. For example, the upper-most licensing module 512 can maintain software, such as an audit tool 530, that tracks use of client applications in underlying computers in the network. An audit report can be periodically generated and sent upstream to the license server 526. The license server 526 can record and interpret the audit report to monitor use of client application software. Alternatively, license validity inquiry request message traffic from individual client applications can be recorded in the license server 526. Audit information can be used to generate billing invoices.

DEPR:
An additional aspect of the aforementioned audit system permits an MIS manager at a corporation or institution to monitor the use of client applications for internal audit purposes. Such a system operates by monitoring license inquiry traffic passing through a network to and from a license server. In an exemplary embodiment, such a system involves maintaining internal auditing software (e.g., a tool or utility program) in an upper-most level licensing module in an internal network. A report can be generated by the internal auditing software tool. Data in the report can be derived from information collected at the upper-most licensing module. The MIS manager can use the internal audit reports to manage the licensing arrangements of the client applications on the network. For example, if a network of twenty desktop computers is frequently using a maximum number of floating licenses for a particular client application, the MIS manager can ascertain this by reviewing internal audit records, and take appropriate action.

DEPR:
In accordance with a preferred embodiment of the invention, the license system operates by distributing licensing information to the cache components 520 in the licensing modules in response to inquiry requests. The information contained in a particular cache component 520 is specific to subordinate software applications 514, or licensing modules. In accordance with a preferred embodiment, license information is organized by class designations. Individual licenses for client applications on desktop machines 500 can be covered by sub-class licenses maintained in the cache components 520 in the file server computers 502. The sub-class licenses on the file server machines 502 can, in turn, fall under a class license maintained in the cache component 520 of the minicomputer 506. The class license maintained on the minicomputer's licensing module can be designated under a block license maintained in a cache component 520 of the main computer 512. The client component 522 of the main computer's licensing module communicates with the license server 526 to verify block licenses.

DEPR:
When a client application is first brought up, the Check Out License procedure is initiated. The purpose of the Check Out License procedure is to enable the software application to which the client module is attached. In addition, the Check Out License procedure can be used by to track the proliferation of a client application. In accordance with an exemplary process incorporating the invention, the client module's Check Out License call generates a client data structure containing: the name of the software application, any feature name(s) that is to be enabled, the name of the upstream agent component/module, a hardware identifier of the computer containing the client application, a date/time stamp, and a version number of the licensing system. Other fields are created in the client data structure that are filled in by the upstream licensing module or license server.

DEPR:
The client module investigates the client data structure content returned by the

agent component/module to analyze the license and authorization ID information contained therein, if any. If these fields indicate the presence of a valid license, the client module enables the software application. The license and authorization IDs are stored in the client data structure in the client module for future license validation checks. If the data structure fields for the license and authorization IDs are null, the client application is not enabled and the client data structure is deleted from the client module.

DEPR:
A licensing system in accordance with the invention, in a preferred embodiment, involves inserting licensing system code into a pre-compiled version of a software application and then compiling that application into a single executable client application. However, in accordance with another embodiment of the invention, the licensing system can be provided as a module that is inserted into to an existing software structure on a computer network. Such a system can be used to monitor software application use in a computer network that does not otherwise have a means to audit application use. This type of system can be used by individuals, such as MIS managers, who wish to audit software application use activity in a network. In an exemplary embodiment, a client module installed in an individual desktop computer "wraps around" software applications selected for monitoring. When the wrapped application used, or seeks a floating license from a file server, a licensing module installed in the file server computer records the activity. In a floating license system, the licensing module can be configured to always reserve a license for use by certain individual computers (e.g., the CEO's computer). Audit records generated by the licensing module can be periodically checked by the MIS manager to see if adjustments are needed (e.g., purchase a greater number of floating licenses).

DEPR:
A variation of the latter system also can be installed in existing network to retrofit a licensing management system. Client modules wrap previously installed applications thereby converting them into client applications. The client modules on individual computers monitor and/or control client application use. The client modules can report to licensing modules in upstream file servers, or report directly to a license server over a public network. Such a system can be used by software providers as an aftermarket component installed on top of existing software systems. In such a setting, a client module can be responsible for handling license validation of more than one client application loaded on the computer. If two or more client software applications on a computer are by a same software provider, the client module can generate a single validity request message covering each wrapped client application. Such a system has the effect of providing a generic licensing validation system for all of the licensed software on a desktop machine supplied by a particular software provider.

DEPR:
A exemplary licensing system that can be retrofitted in an existing computer network is depicted in FIG. 6. An individual computer 600 has a client module 618 installed therein. The client module 618 is wrapped around one or more software applications 620, 622, 624 and 626 to create client applications. In a preferred embodiment, the client applications are specific to the software provider who is retrofitting their networked software with the licensing system. However, if a license record database is configured as a "clearing house," whereby a multitude of software providers consolidate license information in a single server or a network of license servers, the client module 618 can validate software licenses by a variety of software providers. The client module 618 can enable, or validate, the client applications by communicating with a licensing module in a file server 602. The licensing module in the file server 602 also is a retrofitted component in the license system. Licensing modules are installed in each of the computers (e.g., file servers, minicomputer, main computers) that form the network hierarchy. Operation of such a system is substantially similar to that of a network embodiment of a license management system described above. Alternatively, client modules can communicate directly with a license server 604 over a public network, such as the Internet 616.

DEPR:
As mentioned in the Background above, software licensing management systems conventionally maintain proprietary, that is application specific, licensing code in software applications (i.e., application portion). Corresponding proprietary licensing code also is maintained in the file server or like network element

(i.e., authenticator portion). Consequently, a conventional system typically has to maintain several separate licensing validation programs on a file server to check each of the software applications loaded on sub-tended computers. A system in accordance with the latter embodiment of the invention provides for a generic solution whereby a single client module is maintained on the desktop computer that handles all of the licensing management for the computer's client applications. Hence, the computers and corresponding servers need not contain several licensing system applications each having proprietary code.

DEPR:
The encrypted fields of the data structure contents passing between a client and agent include the license and authorization Ids, and any proprietary data required for validation, such as floating license information. All of the fields of the data structure contents can be encoded, however, the application name, feature name, hardware identifier, and a licensing system version number are preferably left unencoded. In the event that there is lack in synchronization, a client or agent can look to the unencoded information and revert back to a most recent encryption key. If there is still lack of synchronization, a request is passed that the client return to the initial message state, thereby allowing both the client and agent encryption engines to reset.

DEPR:
The frequency of validation checks is application dependent. A software designer can select when and how often validation checks are to occur, if at all. The licensing system can be configured in accordance with the needs of a particular application. The software license can be validated, or enabled, each time the application is brought up on a computer, or each time a particular feature is used (e.g., printing). The software license also can be validated in response to the expiration of a timer (i.e., periodic validation). If the response to the validation check is returns a null license ID, the client application is disabled. To prevent a user from tampering with a client application, or a licensing module associated therewith, in an effort to disable validation checking, any number of watchdog timers can be nested in the client application. The nested watchdog timer can be used to periodically self-check the client application to determine whether it has been validated within the watchdog period. If so, the watchdog timer is reset. If not, a validation check can be initiated or the client application can be disabled.

DEPR:
Establishing a database license record in the licensing server can be performed in a variety of ways. Software can be purchased and paid for in an interactive commercial transaction conducted over the Internet, as described above. The result of such a transaction is to establish a license record in the licensing server database. A subsequent validation check by the client application will allow the software to be enabled. The database entry also can be formed by pre-authorization. If the software is purchased from a vendor, the vendor, in the context of the transaction can perform the database entry shortly after the software is supplied to the user or company. When the software is brought up on the client computer, an initial validity check will return an enablement response because a license record has already been established. Alternatively, software can be pre-enabled with a temporary term license thereby providing a software provider with a time window in which to establish a license record. Other techniques for establishing an entry in the database, and thus enabling the corresponding client application, include using automated telephone operator systems. A client can call a telephone number and use a touch-tone phone to respond to prompts presented by an automated operator. Hence, any mechanism for initializing the database, and consequently automatically enabling the software, is deemed suitable.

DEPR:
Another aspect of systems operating in accordance with the invention is feature enablement. The systems described above can be used to enable and disable particular features in a client application. Such a situation may occur wherein a software application has several levels of operating capability. For instance, a user can selectively enhance operating capability by selecting features defined in a software feature application menu. In response, an associated client module can invoke the Check Out License procedure wherein the desired feature name is passed upstream. The license, of course, does not exist yet, but the system can be configured to direct the user to, or provide the user with, a feature

enablement menu that requests that the user enter credit card information, as described above. Alternatively, the system can be organized to automatically initiate a process that creates a license when new software is brought up. This can involve a mechanism that forms a Web server connection and supplies an authorization message that creates a database license entry. For a commercial client, a software provider can monitor the activation and use of client application features and bill the client accordingly.

DEPR:
As previously discussed, systems in accordance with the present invention can be used to audit the use and proliferation of software. Attachment of a licensing module to a software application causes that software application to report back to a licensing server at some point. If the client module or software application is configured to report back the first time that the software application is brought up, a software provider can keep track of, or audit, which and how many machines the software application is operating on. The licensing server can be set to initially enable any request received for the software application being audited. At the discretion of the software provider, the software application can be turned off at anytime. The software provider can respond to subsequent licensing enablement or validation requests by supplying a prompt inviting the user to purchase a license. A variation on the audit function can be used to log questionable user activity.

DEPR:
Software version control can be provided in accordance with another aspect of the invention. Software version number information can be passed upstream in the context of a license validity inquiry request message. If the software version has expired, a message can be passed back in the status field of the authorization ID indicating this information to a user. The user may then be asked to license and procure a new version of the software. Software version control can be used to prevent version collision, and to force users to stop using expired software versions by simply disabling them.

DEPR:
Systems not having network, or Internet access, can still be enabled in an automated, albeit semi-manual fashion. The licensing module in a software package to run on a non-networked machine can contain a tag indicating this fact. The user may be prompted to call a number and receive recorded information for enabling the software. This information, of course, requires initial manual entry to enable the software, and periodic manual entry to validate the software. The validation check can be performed by prompting a user to call a number to automatically receive further enablement instructions. Such instructions are only provided if the license remains valid. Optionally, instructions can be automatically be returned by fax in response to a validation request phone call.

DEPR:
The invention has been described with respect to several exemplary embodiments. However, one skilled in the art will readily appreciate and recognize that the licensing system or method of operation in accordance with the invention can be applied in any computing system using licensed software, which systems are preferably attachable to a public network, such as the Internet. The invention provides the benefit of being able to freely distribute licensed software incorporating the invention with reduced apprehension of the software being illicitly copied or used without its being properly licensed. Alternatively, a system in accordance with the invention can be used to track and maintain records of the proliferation and use of software incorporating the invention.

CLPR:
1. A network software licensing system having self-enabling software, the network licensing system comprising:

CLPR:
3. The network software system claimed in claim 1, wherein license records are organized according to class and sub-class designations.

CLPR:
7. A hierarchical license system having a plurality of computers connected by a computer network, the plurality of computers arranged in a hierarchy from a lowest level to highest level and having computer software that is enabled with a

license, the system comprising:

CLPR:
10. The management licensing system claimed in claim 8, wherein the library of code modules further includes:

CLPR:
12. In a computer system having a plurality of computers connected by a computer network, the plurality of computers arranged in a hierarchy from a lowest level to highest level and having computer software that is enabled with a license, a method of enabling the computer software with a license, the method comprising the following steps:

CLPR:
22. A hierarchical computer software licensing apparatus, the apparatus comprising:

CLPV:
license server on a computer at the highest level in the hierarchcial license system, the license server having a database of licensing information for enabling computer software on the plurality of computers in the hierarchcial license system;

CLPV:
an interupt validation generator that initiates a license validity inquiry when a piece of the computer software into which the interrupt validation generator is inserted is executed.

CLPV:
a client application comprising a licensed software application and a portable client module; and

CLPV:
(a) maintaining a license server on a computer at the highest level in the hierarchical license system, the license server having a database of licensing information for enabling computer software on the plurality of computers in the hierarchy;

CLPV:
(b) requesting a license in a licensing module at a level L in the hierarchy from a licensing module on a computer at level L+1 in the hierarchy to enable computer software on a computer at a level L in the hierarchy, the licensing modules having:

CLPV:
maintaining means for maintaining a license server on a computer at a highest level in a hierarchical license system, the license server having a database of licensing information for enabling computer software on a plurality of computers in the hierarchy;

CLPV:
requesting means for requesting a license from a computer at a level L in the hierarchy to enable computer software on the computer from a licensing module on a computer at level L+1 in the hierarchy, the licensing modules having:

CLPW:
cache component for storing license information for computer software on computers in a previous level L-1 in the hierarchy, wherein the cache component is updated periodically with license information from agent components in a next level L+1 in the hierarchy,

CLPW:
wherein a client component in a licensing module at a level L in the hierarchy requests a license for enabling software from an agent component in a licensing module at a level L+1 in the hierarchy and not directly from the license server at the highest level in the hierarchy.

CLPW:
cache component for storing license information for computer software on

computers in a previous level L-1 in the hierarchy, wherein the cache component is updated periodically with license information from agent components in a next level L+1 in the hierarchy;

CLPW:
cache component for storing license information for computer software on computers in a previous level L-1 in the hierarchy, wherein the cache component is updated periodically with license information from agent components in a next level L+1 in the hierarchy;

CLPX:
the client component in the licensing module systematically initiates communication with the agent module in the license server over a public network to collect license records for storage in the cache component; and wherein the client module automatically initiates communication with the agent component in the licensing module at a level in the hierarchy above the client module and not the agent component in the license server to determine whether the cache component in the licensing module contains a license record corresponding to the software application.

WEST

Generate Collection

L16: Entry 18 of 26 File: USPT May 26, 1998

DOCUMENT-IDENTIFIER: US 5758068 A
TITLE: Method and apparatus for software license management

ABPL:
A license key with method of implementation is disclosed. The license key is used for accessing a licensed product on an enterprise computer system. Typically, the enterprise computer system includes a plurality of computer systems, but the computer systems are not necessarily interconnected. A first identifier code from the enterprise computer system, such as the system serial number, is used for allowing the license key to identify the enterprise. A second identifier code from a computer within the enterprise system is used, which is typically a serial number from the computer to identify it to the license key as being part of the enterprise system. Lastly, a third identifier code is used, which is selected from the licensed product to be used on the enterprise system and is tied to the enterprise system number, thereby allowing the licensed program to be accessed on the enterprise system with only a single key, irrespective of the number of computers accessing the licensed program. The setup of the license key, along with calling the identifier codes from the enterprise system, each computer, and the license product, is performed by a license manager, which also provides other services based upon the license key. The license manager provides either usage control, access control, or both, based upon information generated by the license key and found within a product information packet provided with the licensed product and based upon the enterprise system.

BSPR:
The present invention relates to data processing systems and, more particularly, to a management scheme for software licenses operating on the data processing systems.

BSPR:
Current state of the art software license protection methods support software license enforcement based on a "license key" concept. A license key, herein referenced as "key," is an encrypted string that contains information such as, for example, the software product's ID, the software product's entitled usage limit and expiration date, and the serial number of the system where the key may be installed. Computer enterprises, which consist of multiple systems, are supported using either of the two following methods.

BSPR:
Firstly, a key is created, then distributed, then installed, for each licensed software product on each system. Unfortunately, this method can result in a large number of keys that are often quite difficult to manage. For example, an enterprise with 11,000 systems and three software packages would need 33,000 unique keys. In order to reduce the number of keys, some vendors offer a key that is not based on a system serial number and that can be installed on any system. For this example, only three keys would then be needed, meaning one per licensed product. Unfortunately, this approach offers no asset protection at all since the key can literally unlock the software to run on any system in the world.

BSPR:
Another issue facing software licensing management is that of enforcing both usage control and access control via software keys. All current key-based license management methods support either usage control or access control, but not both. This requires that both developers and customers must interface to two disparate methods.

BSPR:
In usage control, a software product protects itself at run time by making calls to a licensed manager that accesses a stored license key to see if the terms and conditions of the license are being met. For example, if only four (4) current users are allowed then the license manager tells the software product when the fourth user has signed on. This form of protection is common for usage-based pricing.

BSPR:
Accordingly, what is needed is a software licensing method for use on an enterprise system that provides an enterprise key that avoids the complexity associated with other mechanisms or that does not become unduly simple to override or access. Additionally, what is needed is a way of providing access control without being solely dependent upon one control protocol or the information manager as is required in current systems. Further, what is needed is a method that simplifies the management of grace periods without burdensome and inconsistent protocols.

BSPR:
It is another object of the present invention to provide a management scheme for software licenses operating on the data processing systems.

BSPR:
It is yet another object of the present invention to provide the enforcement of software licenses across all systems within an enterprise.

BSPR:
Dorothy Boudreaux at RN.com
The foregoing objects are achieved as is now described. According to the present invention, a license key with method of implementation is disclosed. The license key is used for accessing a licensed product on an enterprise computer system. Typically, the enterprise computer system includes a plurality of computer systems, but the computer systems are not necessarily interconnected. A first identifier code from the enterprise computer system, such as the system serial number, is used for allowing the license key to identify the enterprise. A second identifier code from a computer within the enterprise system is used, which is typically a serial number from the computer to identify it to the license key as being part of the enterprise system. Lastly, a third identifier code is used, which is selected from the licensed product to be used on the enterprise system and is tied to the enterprise system number, thereby allowing the licensed program to be accessed on the enterprise system with only a single key, irrespective of the number of computers accessing the licensed program.

BSPR:
The setup of the license key, along with calling the identifier codes from the enterprise system, each computer, and the license product, is performed by a license manager, which also provides other services based upon the license key. The license manager provides either usage control, access control, or both, based upon information generated by the license key and found within a product information packet provided with the licensed product and based upon the enterprise system.

BSPR:
The access control may include encrypting data within the licensed program that can only be decrypted with the license key. The encrypted data is only a sparse portion of the license program. The usage control may include a grace period that, upon expiration, is only extendable by way of the license key.

DRPR:
FIG. 5 depicts the relationship of license information to the license manager and to the licensed programs.

DRPR:
FIG. 6 depicts a prompt for a command that enables a vendor to package license information within a licensed program.

DRPR:
FIG. 7 depicts a prompt for a command that enables a vendor to create a key for a licensed program.

DEPR:
Next, for each licensed program (LP) covered by an enterprise licensing agreement, a key is issued that ties that particular LP to that enterprise's ID. For example, if an enterprise needed licenses for three LPs on 1000 systems, it would need only one key for each LP (instead of the 1000 for each LP if each key was based on a unique system serial number) with each key based on the enterprise ID.

DEPR:
In order to direct the license manager in each system to accept enterprise ID-based keys, a Software License Enterprise Enabler Program (SLEEP) 40 is provided. As shown in FIG. 2, the SLEEP 40 itself is key protected and will only run in the presence of a sleep key 42 that ties it to a particular system. When SLEEP 40 runs on that system, it creates an Enterprise Key based on the serial number of the system it runs on and on the enterprise ID. This key 42 can then be installed on any system in the enterprise and its presence on any system will indicate to the license manager 44 that licenses for other LPs (based on that Enterprise key) should also be honored. Another way of looking at this design is to say that license manager 44 has dormant (or `sleeping`) support for enterprise licensing and only needs SLEEP-created key 42 to unlock that support.

DEPR:
The actual use of enterprise key 42 by license manager 44 is depicted in the flowchart of FIG. 3. License manager 44 in step 310, receives a request to use the license for a licensed program for a (LP). Next, in step 312, the system searches for keys 42 for the licensed program that are based on a local system serial number. Next, in step 314 the system determines whether a valid key has been found, and if so, in step 316 returns an okay status. Otherwise, if no valid key is found, the system, in step 318 searches for the enterprise key. If the system finds the enterprise key, as depicted at step 320, the system proceeds to step 322; otherwise, if no enterprise key is found, the system sends an error status of no key found in step 324 as illustrated at reference numeral 46 in FIG. 2. In step 322, the system searches for keys for the licensed program that are based on the serial number in the enterprise key. In step 326, the system determines if a valid key has been found and, if so, in step 328 returns an okay status of "key found." If no valid key has been found, the system in step 330 return an error status of "no key found."

DEPR:
Usage control is where the operating system protects itself by making calls to a license manager to determine if the software is running on the correct system or within certain usage limits as determined by a license key, or both.

DEPR:
Access control is where the data on the distribution media are encrypted and can only be decrypted by "installing" software via information in a license key. Access control is often used for non-executing software, such as priced on-line books, fonts, and dictionaries.

DEPR:
To provide access control for licensed programs, the licensed programs are packaged according to a system management services (SMS) packaging method used in the operating system of the enterprise in FIG. 1. The SMS packaging is required of licensed programs in order to make them installable and serviceable. The block diagram in FIG. 5 further illustrates the hierarchy of the operating system including SMS packaging method. To begin, a licensed product 62 consists of various operating system objects 64 such as programs, files and commands. The developer ties these objects together into a package via a product packaging information routine 66, which is one of the system objects of the operating system. The packaging information 64 describes the product name, ID, options, and other information needed to be able to install, service, and protect the licensed product. The packaging information is created via commands shown in routine 66. The AddLicInf command defines basic licensing information such as the vendor password, which is used in validating keys, and whether it is usage controlled, access controlled, or both. The ChgPrdObjD (Change Product Object Description) is used to mark objects for inclusion in the package. This command is also modified to mark which objects are to be accessed protected. For performance reasons, it is not reasonable to protect all the objects in a package, so the developer marks just enough objects to protect the product. Next, the save license program

(SavLicPgm) command 70 writes the product to media 72 in the appropriate operating systems save/restore format. If the developer specifies encryption, via a parameter in SavLicPgm, then those objects marked in the previous step are decrypted just prior to writing them to the media. The product packaging information object, however, is then written to the media in non-encrypted form and "access header" file is then written to media 72 as an indicator of what objects were decrypted.

DEPR:
Once the product packaging information object has been written to media 72, license key manager 74 then generates a key, up to N keys, for the number of licensed programs that are used in the system. Add License key (AddLickey) program 76 performs the keying operation which is based on the restored licensed program (RstLicPgm) 78 sent from media 72.

DEPR:
Two more commands used during installation, as shown in FIG. 5 as well, are Save License Program (SavLicPgm) 70 and Restore License Program (RstLicPgm) 78. The Save License Program 70 includes an access protect parameter, that if set to yes, allows those objects marked for access control to be decrypted prior to writing them to media 72. This command also writes "access header" file to the media. This file contains a list of the objects that were decrypted. And the list is then used by the Restore License Program to determine which objects need to be encrypted at installation time.

DEPR:
The Restore License Program (RstLicPgm) 78 command restores the product packaging information object and the "access header" file from the media. Next, the program then determines which objects in the product need to be encrypted prior to restorating or installation.

DEPR:
The next area of concern addressed by the present invention is that of providing grace period support. The license management functionality further supports a vendor specified grace period for the situation where there is no key and the usage limit has been exceeded. The vendor need only indicate the grace period within the application vital product data and monitor for messages or return codes for the license manager. All counting, reporting, administration is handled by the license manager. This is easily integrated by the application developer since it is already required for the key based license management application described above. FIG. 8 is a block diagram illustrating the grace period features according to the present invention.

CLPR:
1. A method for software license management for a licensed program within an enterprise computer system which includes a plurality of computers, said method comprising the steps of:

CLPR:
2. The method for software license management according to claim 1, wherein said step generating a third identifier code associated with said license program further comprises the step of utilizing at least a portion of a serial number associated with said licensed program to generate said third identifier code.

CLPR:
3. A method for software license management for a license program within an enterprise computer system according to claim 1, wherein said step of controlling the usage of said licensed program at any point within said enterprise computer system utilizing either said first identifier code in combination with said third identifier code or said second identifier code in combination with said third identifier code further comprises the step of providing a grace period of use that, upon expiration, is extendable.

CLPR:
4. A computer program product for software license management for a licensed program within an enterprise computer system, said computer program product embodied within computer-readable media adapted to be utilized within a computer, said computer program product comprising: